Prolog Programming Assignment #1:

Various Computations

Thomas Moskal

CSC 344 – Professor Graci

Due date: 3/15/22

Learning Abstract:

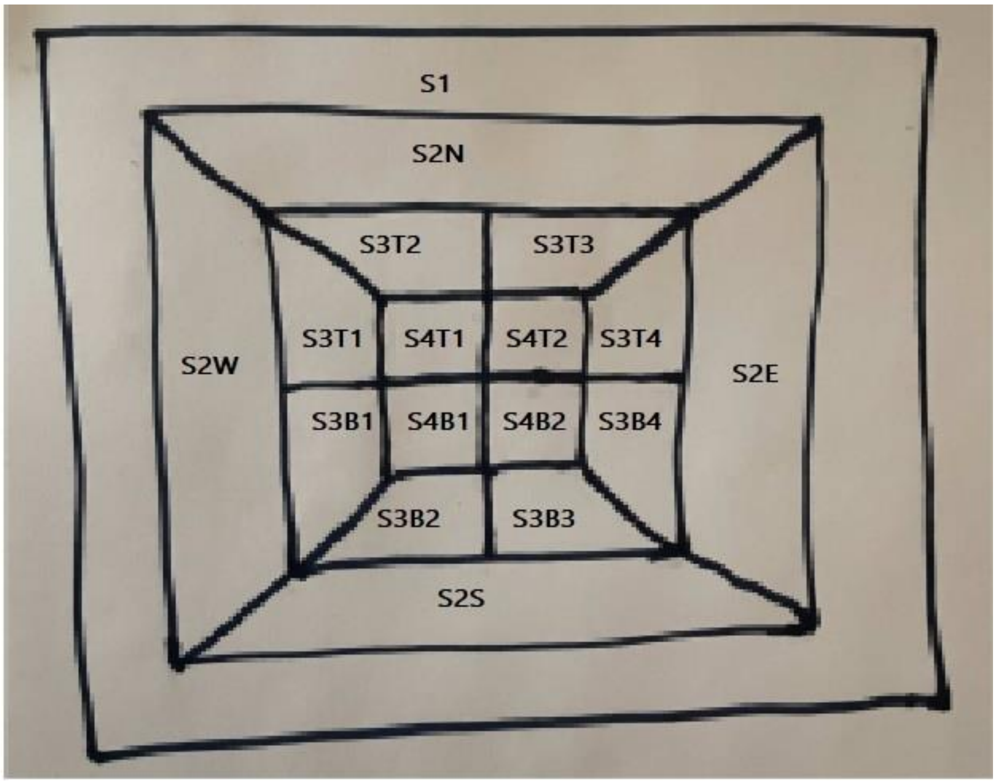
This assignment is our first Prolog assignment. Prolog is a logic programming language. It first appeared 50 years ago in 1972. If you try to halt the program in all caps (HALT.) there is a Hitchhikers Guide to the Galaxy reference. In our assignment, we unfortunately do not see that reference. But! We do a 4-color problem program in the first Task. This is a type of problem where no space can touch an adjacent space with the same color.

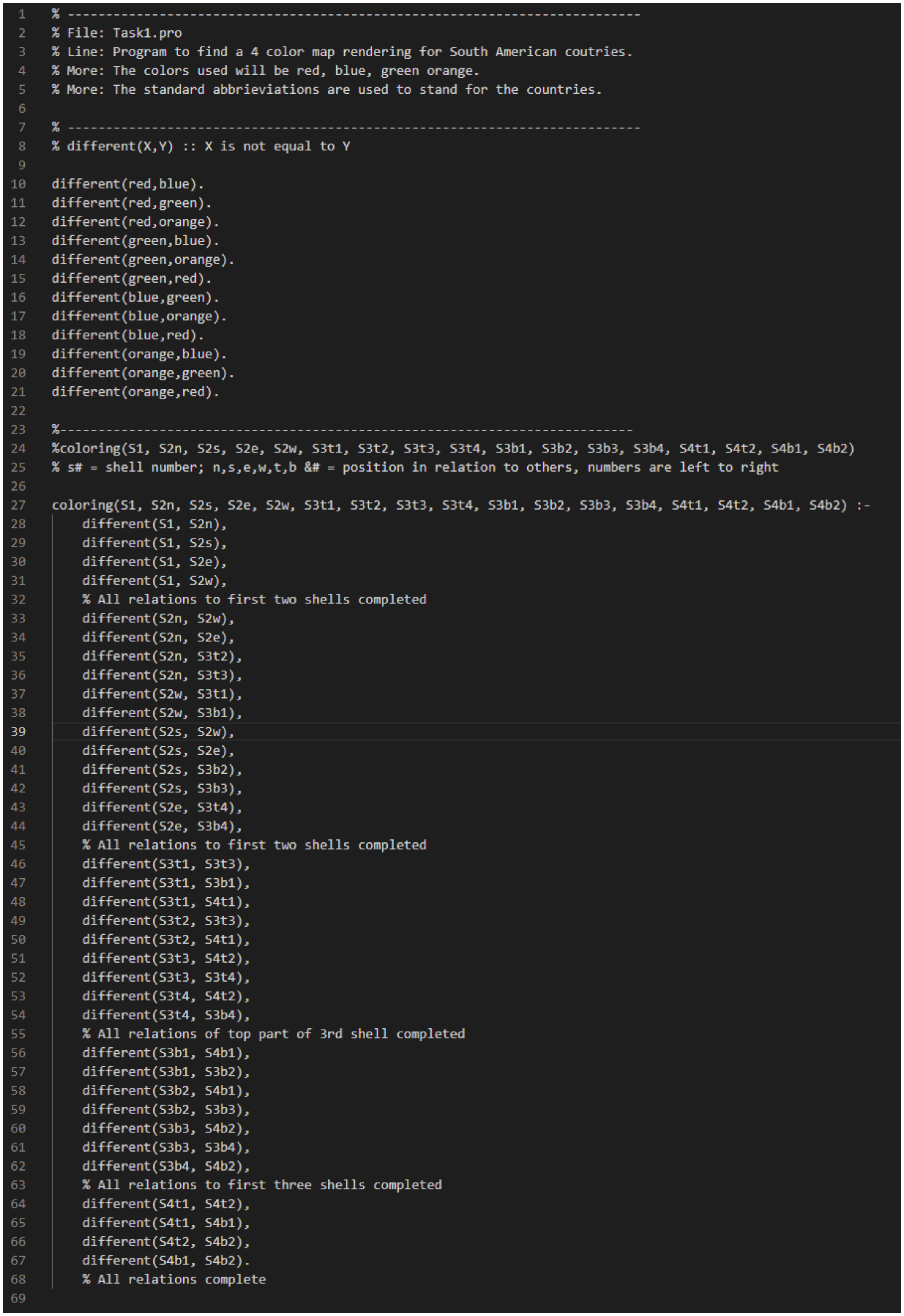
In the second task we do a demonstration in Prolog to examine how we can relate a knowledge base to a set of shapes. This task shows some fundamental prolog code.

The third task involves a knowledge base of several first generation of Pokémon. This task is a good show in how you can pull information about a knowledge base and why that may be useful. Here we can filter Pokémon based on their damage or health and to an astute Pokémon player they might like a database where they can query those stats easily.

Finally, the fourth task involves a lot of list processing, which we know from our time in Racket is a very useful set of skills to have for coding in general.

Task 1 – Map Coloring

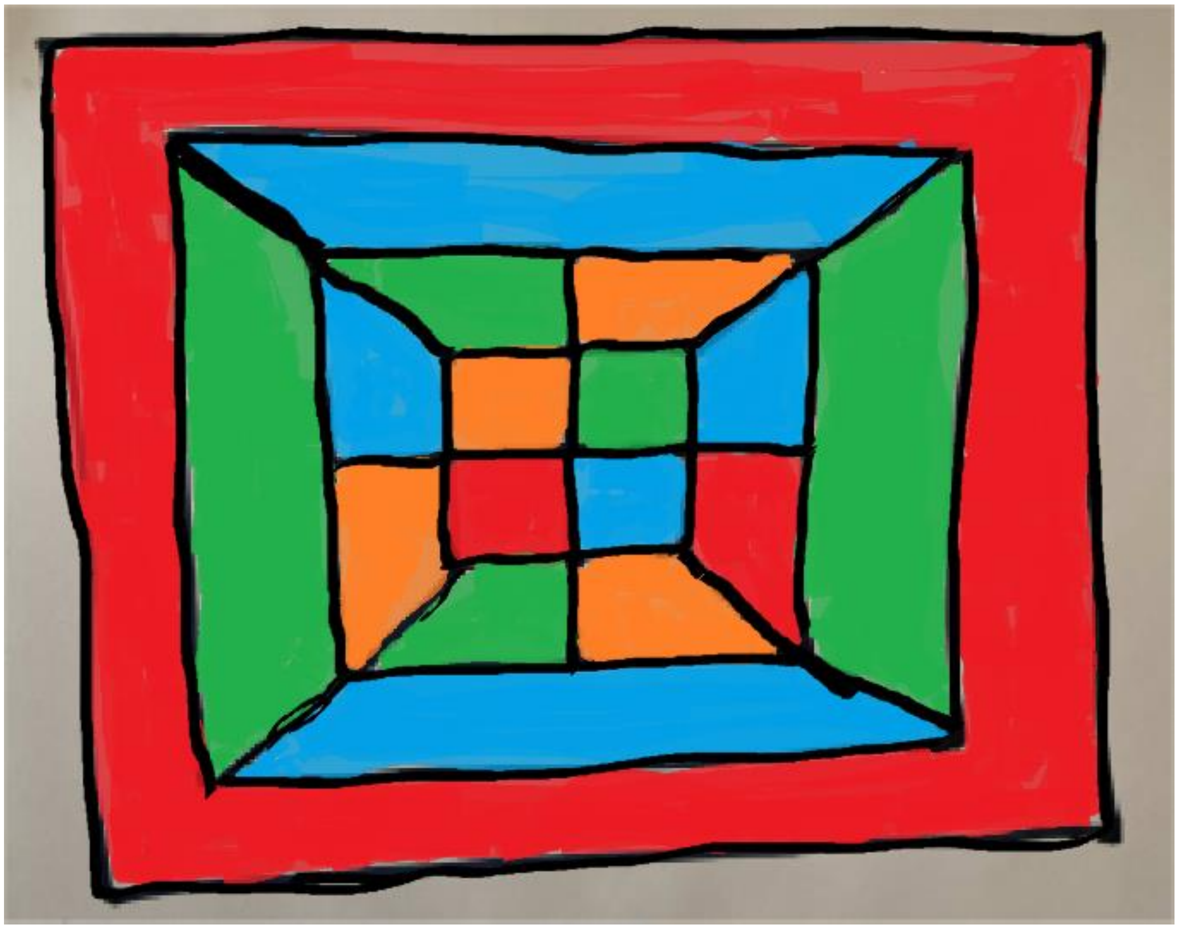




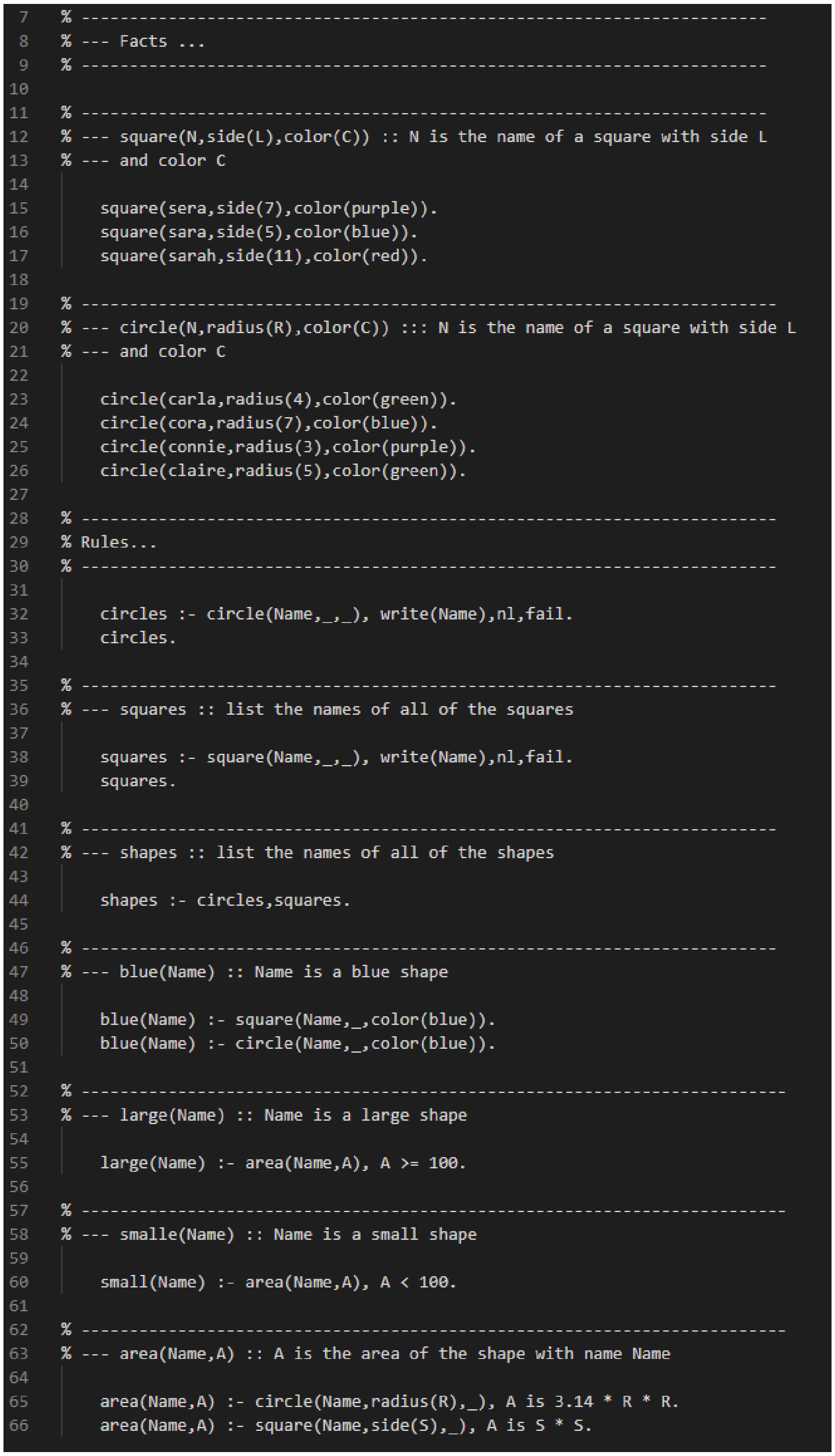
```prolog
% -------------------------------------------------------------------------
% File: Task1.pro
% Line: Program to find a 4 color map rendering for South American coutries.
% More: The colors used will be red, blue, green orange.
% More: The standard abbrieviations are used to stand for the countries.

% -------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y

different(red,blue).
different(red,green).
different(red,orange).
different(green,blue).
different(green,orange).
different(green,red).
different(blue,green).
different(blue,orange).
different(blue,red).
different(orange,blue).
different(orange,green).
different(orange,red).

%-------------------------------------------------------------------------
%coloring(S1, S2n, S2s, S2e, S2w, S3t1, S3t2, S3t3, S3t4, S3b1, S3b2, S3b3, S3b4, S4t1, S4t2, S4b1, S4b2)
% s# = shell number; n,s,e,w,t,b &# = position in relation to others, numbers are left to right

coloring(S1, S2n, S2s, S2e, S2w, S3t1, S3t2, S3t3, S3t4, S3b1, S3b2, S3b3, S3b4, S4t1, S4t2, S4b1, S4b2) :-
    different(S1, S2n),
    different(S1, S2s),
    different(S1, S2e),
    different(S1, S2w),
    % All relations to first two shells completed
    different(S2n, S2w),
    different(S2n, S2e),
    different(S2n, S3t2),
    different(S2n, S3t3),
    different(S2w, S3t1),
    different(S2w, S3b1),
    different(S2s, S2w),
    different(S2s, S2e),
    different(S2s, S3b2),
    different(S2s, S3b3),
    different(S2e, S3t4),
    different(S2e, S3b4),
    % All relations to first two shells completed
    different(S3t1, S3t3),
    different(S3t1, S3b1),
    different(S3t1, S4t1),
    different(S3t2, S3t3),
    different(S3t2, S4t1),
    different(S3t3, S4t2),
    different(S3t3, S3t4),
    different(S3t4, S4t2),
    different(S3t4, S3b4),
    % All relations of top part of 3rd shell completed
    different(S3b1, S4b1),
    different(S3b1, S3b2),
    different(S3b2, S4b1),
    different(S3b2, S3b3),
    different(S3b3, S4b2),
    different(S3b3, S3b4),
    different(S3b4, S4b2),
    % All relations to first three shells completed
    different(S4t1, S4t2),
    different(S4t1, S4b1),
    different(S4t2, S4b2),
    different(S4b1, S4b2).
    % All relations complete
```

```
?-
% c:/Users/Habor/OneDrive/Desktop/CS classes/Spring 2022/CSC_344/Prolog/Assignment 1/Task1.pro compiled 0.00 sec, 0 clauses
?-
|    coloring(S1, S2n, S2s, S2e, S2w, S3t1, S3t2, S3t3, S3t4, S3b1, S3b2, S3b3, S3b4, S4t1, S4t2, S4b1, S4b2).
S1 = S3b4, S3b4 = S4b1, S4b1 = red,
S2n = S2s, S2s = S3t1, S3t1 = S3t4, S3t4 = S4b2, S4b2 = blue,
S2e = S2w, S2w = S3t2, S3t2 = S3b2, S3b2 = S4t2, S4t2 = green,
S3t3 = S3b1, S3b1 = S3b3, S3b3 = S4t1, S4t1 = orange .
```

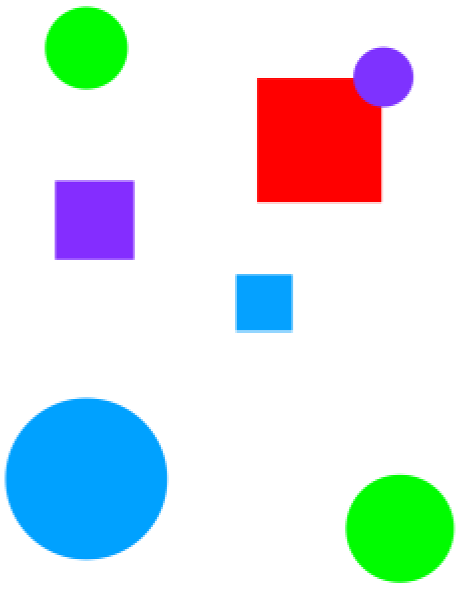
Solution Code and Map

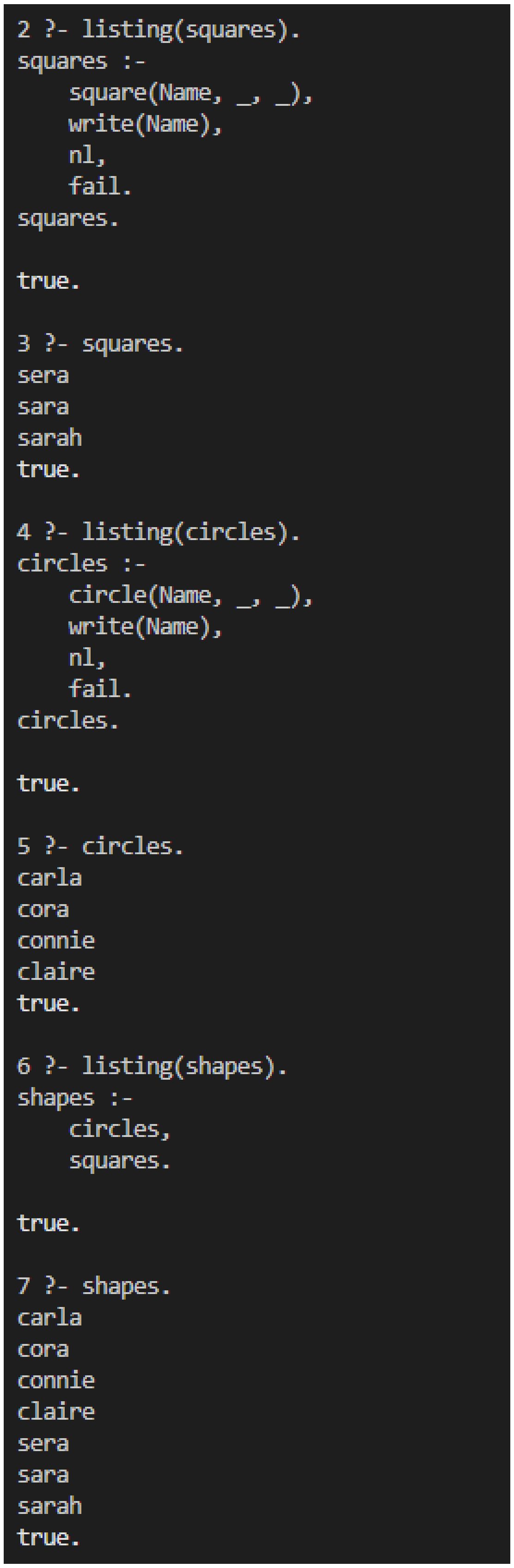


Task 2 – The Floating Shapes World

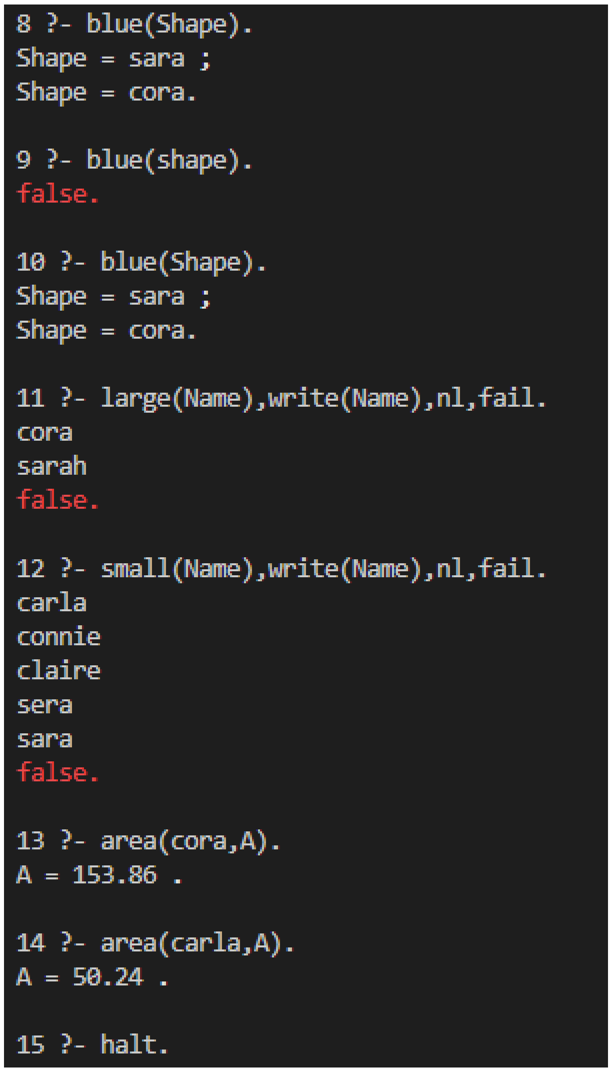


```
% ----------------------------------------------------------------------
% --- Facts ...
% ----------------------------------------------------------------------

% ----------------------------------------------------------------------
% --- square(N,side(L),color(C)) :: N is the name of a square with side L
% --- and color C

    square(sera,side(7),color(purple)).
    square(sara,side(5),color(blue)).
    square(sarah,side(11),color(red)).

% -----------------------------------------------------------------------
% --- circle(N,radius(R),color(C)) ::: N is the name of a square with side L
% --- and color C

    circle(carla,radius(4),color(green)).
    circle(cora,radius(7),color(blue)).
    circle(connie,radius(3),color(purple)).
    circle(claire,radius(5),color(green)).

% -----------------------------------------------------------------------
% Rules...
% -----------------------------------------------------------------------

    circles :- circle(Name,_,_), write(Name),nl,fail.
    circles.

% -----------------------------------------------------------------------
% --- squares :: list the names of all of the squares

    squares :- square(Name,_,_), write(Name),nl,fail.
    squares.

% -----------------------------------------------------------------------
% --- shapes :: list the names of all of the shapes

    shapes :- circles,squares.

% -----------------------------------------------------------------------
% --- blue(Name) :: Name is a blue shape

    blue(Name) :- square(Name,_,color(blue)).
    blue(Name) :- circle(Name,_,color(blue)).

% ------------------------------------------------------------------------
% --- large(Name) :: Name is a large shape

    large(Name) :- area(Name,A), A >= 100.

% ------------------------------------------------------------------------
% --- smalle(Name) :: Name is a small shape

    small(Name) :- area(Name,A), A < 100.

% ------------------------------------------------------------------------
% --- area(Name,A) :: A is the area of the shape with name Name

    area(Name,A) :- circle(Name,radius(R),_), A is 3.14 * R * R.
    area(Name,A) :- square(Name,side(S),_), A is S * S.
```



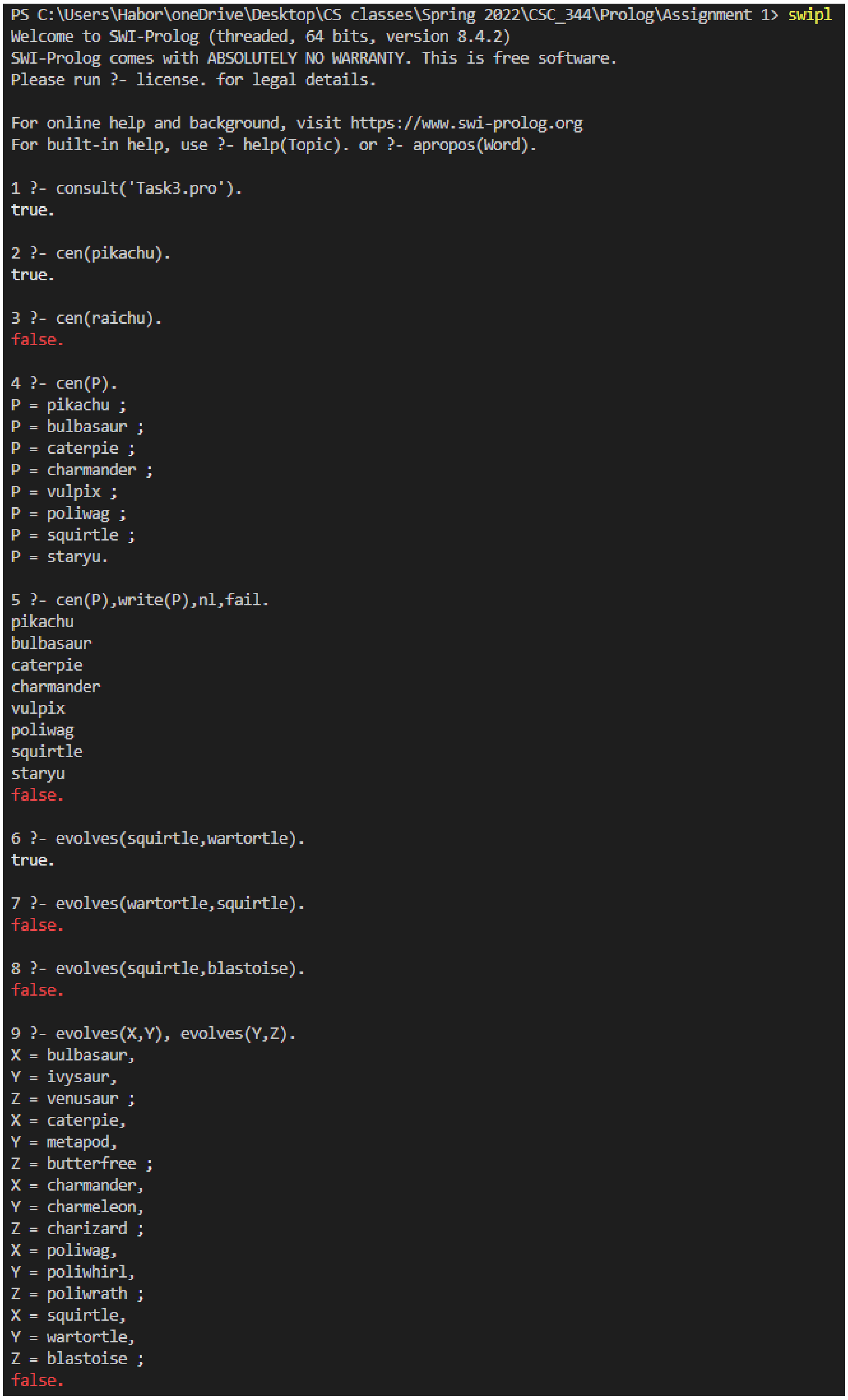


```
2 ?- listing(squares).
squares :-
    square(Name, _, _),
    write(Name),
    nl,
    fail.
squares.

true.

3 ?- squares.
sera
sara
sarah
true.

4 ?- listing(circles).
circles :-
    circle(Name, _, _),
    write(Name),
    nl,
    fail.
circles.

true.

5 ?- circles.
carla
cora
connie
claire
true.

6 ?- listing(shapes).
shapes :-
    circles,
    squares.

true.

7 ?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.
```

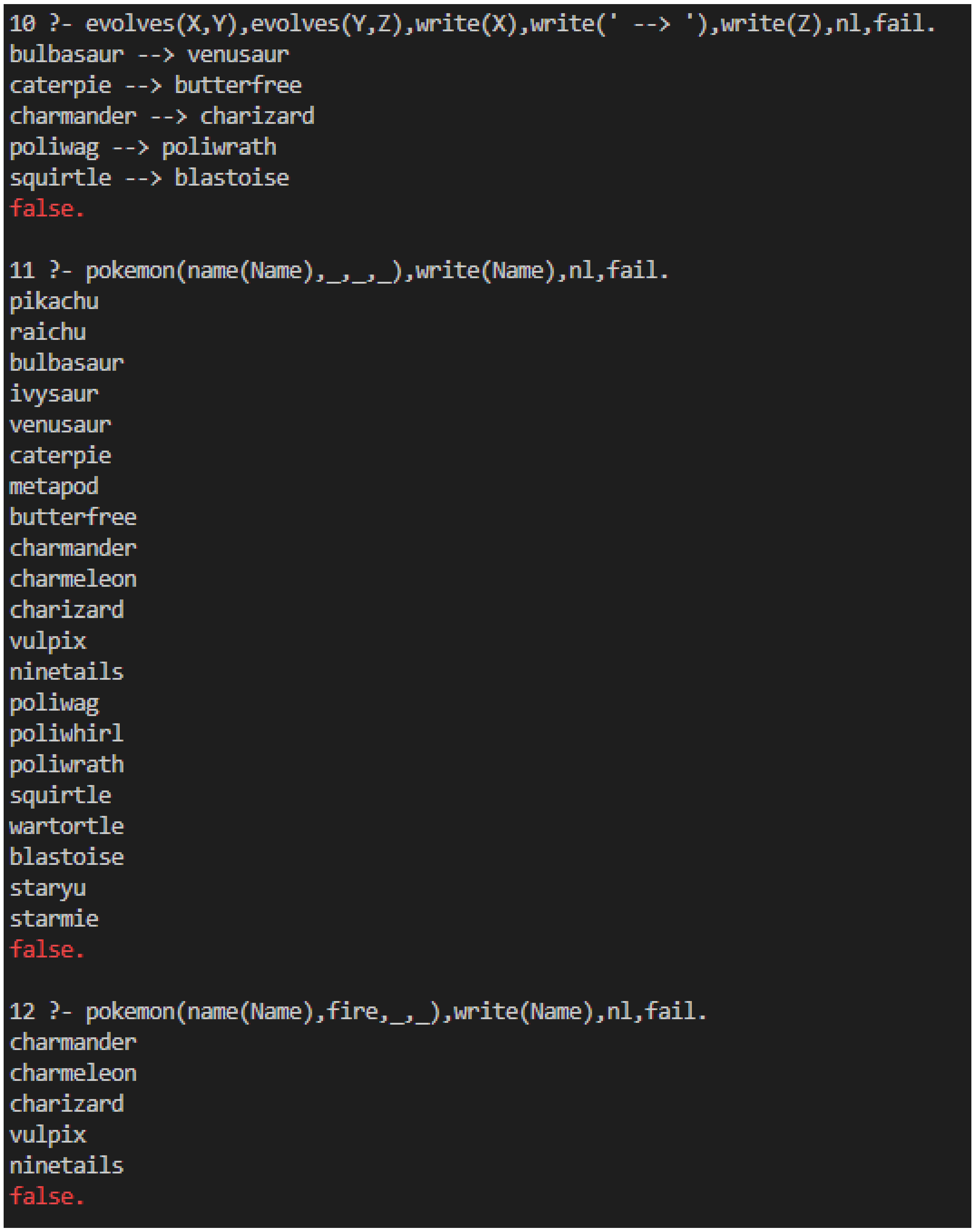


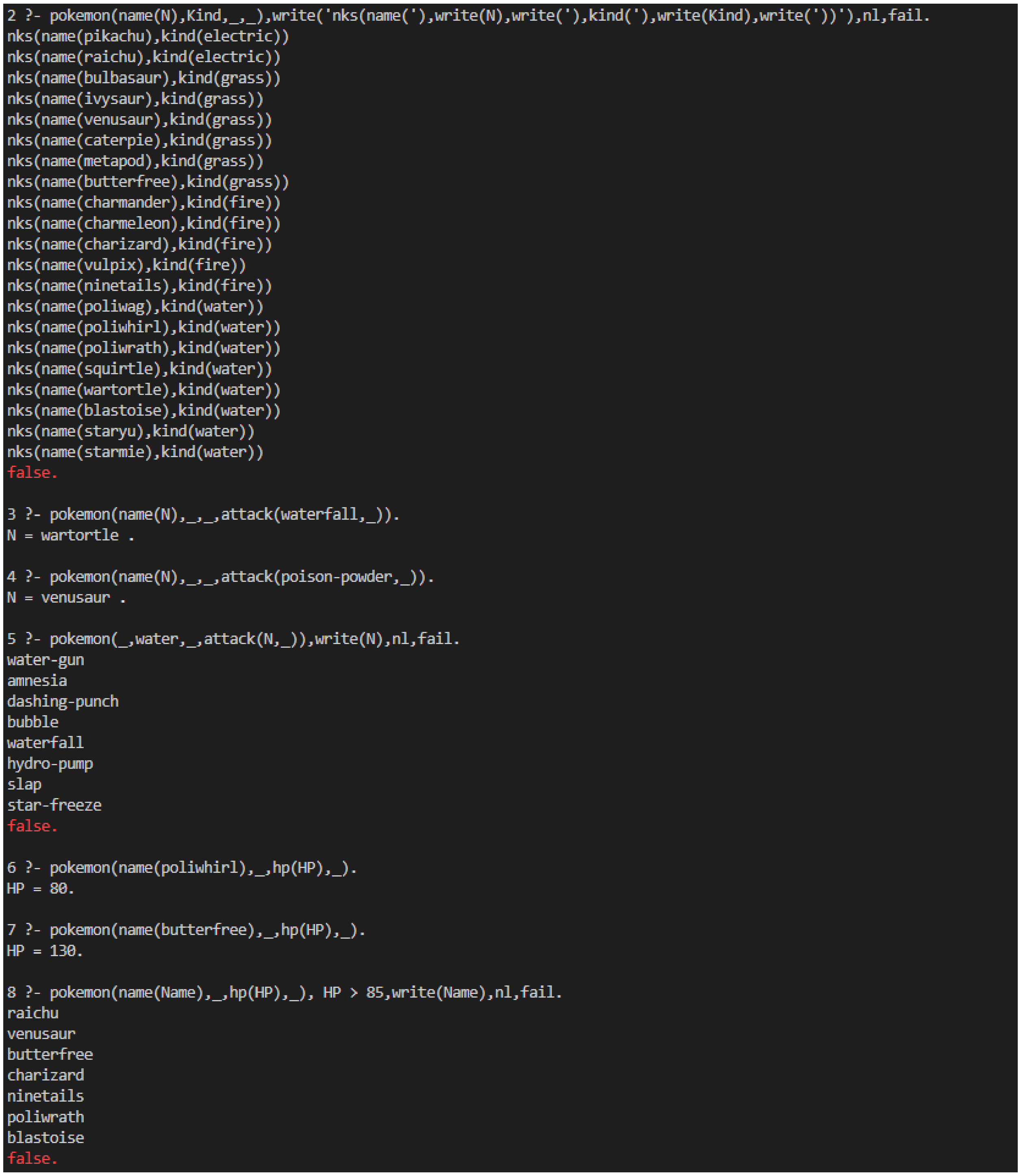
```
8 ?- blue(Shape).
Shape = sara ;
Shape = cora.

9 ?- blue(shape).
false.

10 ?- blue(Shape).
Shape = sara ;
Shape = cora.

11 ?- large(Name),write(Name),nl,fail.
cora
sarah
false.

12 ?- small(Name),write(Name),nl,fail.
carla
connie
claire
sera
sara
false.

13 ?- area(cora,A).
A = 153.86 .

14 ?- area(carla,A).
A = 50.24 .

15 ?- halt.
```

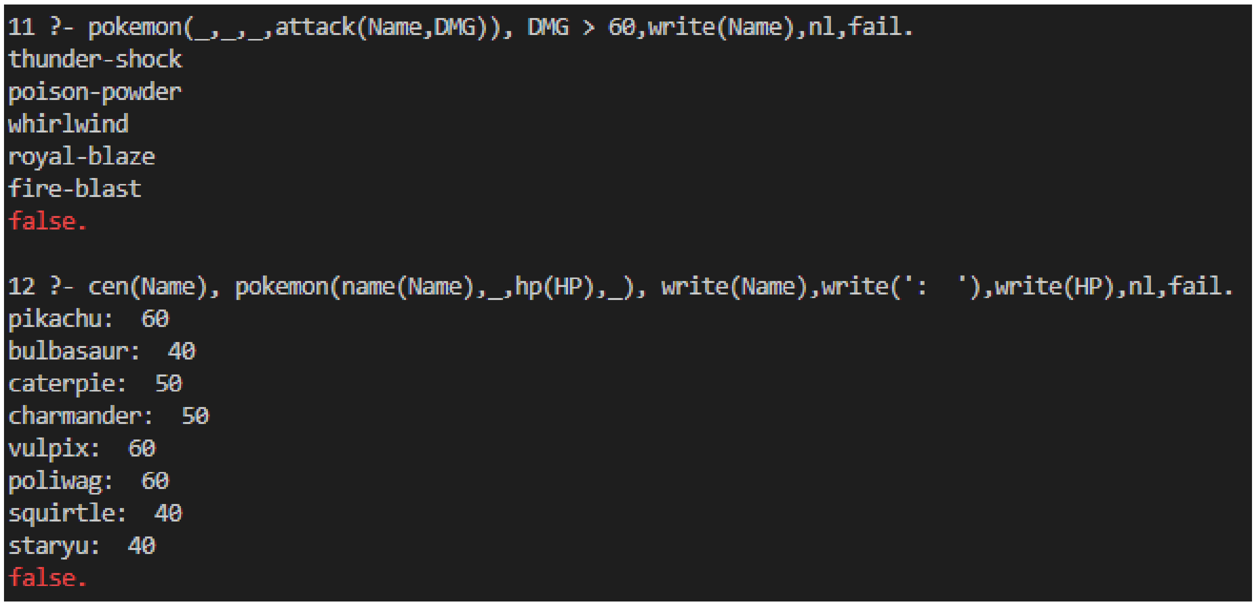
<u>Task 3 – Pokémon KB Interaction and Programming</u>



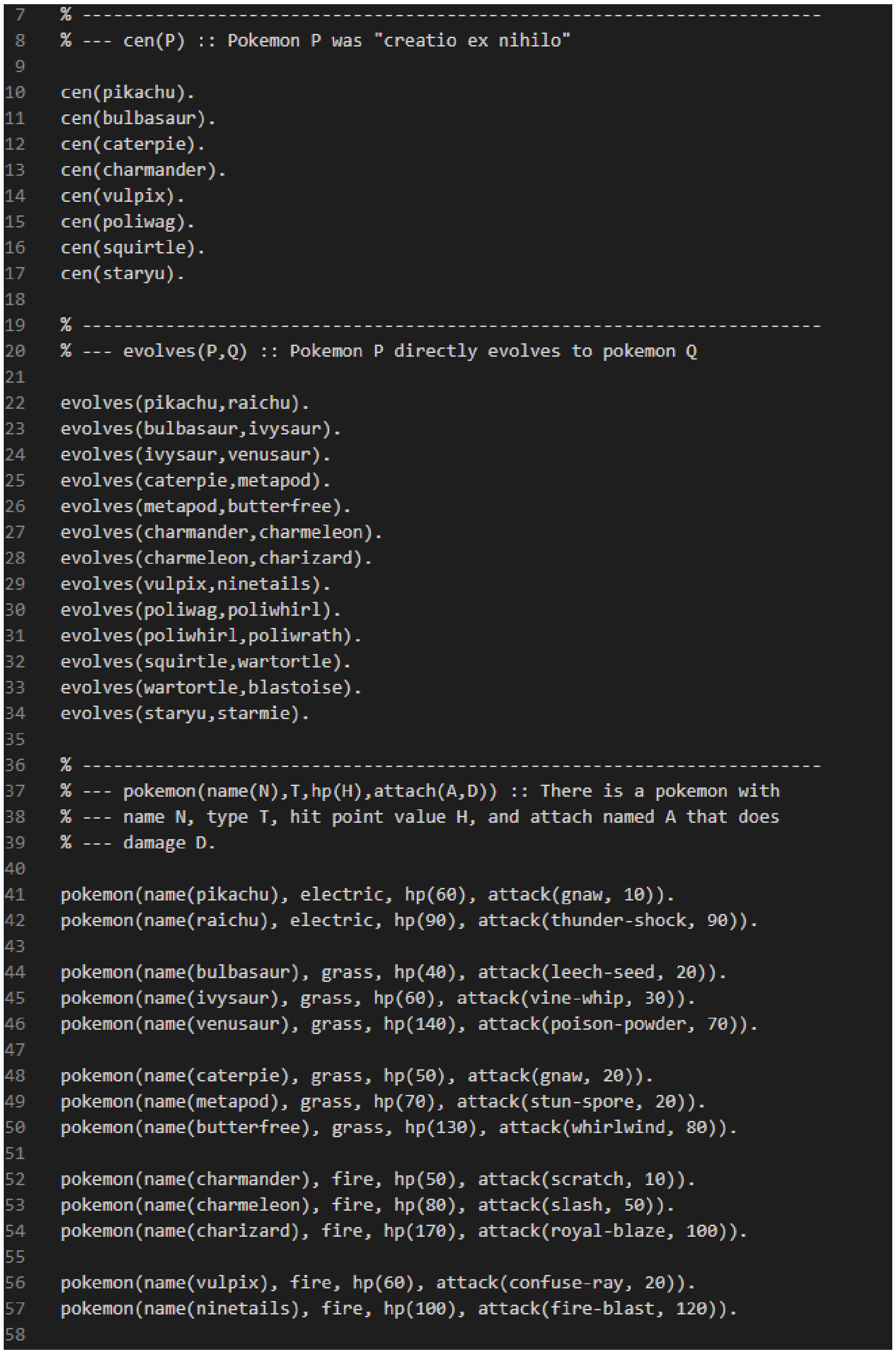
```
PS C:\Users\Habor\oneDrive\Desktop\CS classes\Spring 2022\CSC_344\Prolog\Assignment 1> swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- consult('Task3.pro').
true.

2 ?- cen(pikachu).
true.

3 ?- cen(raichu).
false.

4 ?- cen(P).
P = pikachu ;
P = bulbasaur ;
P = caterpie ;
P = charmander ;
P = vulpix ;
P = poliwag ;
P = squirtle ;
P = staryu.

5 ?- cen(P),write(P),nl,fail.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

6 ?- evolves(squirtle,wartortle).
true.

7 ?- evolves(wartortle,squirtle).
false.

8 ?- evolves(squirtle,blastoise).
false.

9 ?- evolves(X,Y), evolves(Y,Z).
X = bulbasaur,
Y = ivysaur,
Z = venusaur ;
X = caterpie,
Y = metapod,
Z = butterfree ;
X = charmander,
Y = charmeleon,
Z = charizard ;
X = poliwag,
Y = poliwhirl,
Z = poliwrath ;
X = squirtle,
Y = wartortle,
Z = blastoise ;
false.
```


```
10 ?- evolves(X,Y),evolves(Y,Z),write(X),write(' --> '),write(Z),nl,fail.
bulbasaur --> venusaur
caterpie --> butterfree
charmander --> charizard
poliwag --> poliwrath
squirtle --> blastoise
false.

11 ?- pokemon(name(Name),_,_,_),write(Name),nl,fail.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

12 ?- pokemon(name(Name),fire,_,_),write(Name),nl,fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.
```


```
2 ?- pokemon(name(N),Kind,_,_),write('nks(name('),write(N),write('),kind('),write(Kind),write('))'),nl,fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
nks(name(metapod),kind(grass))
nks(name(butterfree),kind(grass))
nks(name(charmander),kind(fire))
nks(name(charmeleon),kind(fire))
nks(name(charizard),kind(fire))
nks(name(vulpix),kind(fire))
nks(name(ninetails),kind(fire))
nks(name(poliwag),kind(water))
nks(name(poliwhirl),kind(water))
nks(name(poliwrath),kind(water))
nks(name(squirtle),kind(water))
nks(name(wartortle),kind(water))
nks(name(blastoise),kind(water))
nks(name(staryu),kind(water))
nks(name(starmie),kind(water))
false.

3 ?- pokemon(name(N),_,_,attack(waterfall,_)).
N = wartortle .

4 ?- pokemon(name(N),_,_,attack(poison-powder,_)).
N = venusaur .

5 ?- pokemon(_,water,_,attack(N,_)),write(N),nl,fail.
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
false.

6 ?- pokemon(name(poliwhirl),_,hp(HP),_).
HP = 80.

7 ?- pokemon(name(butterfree),_,hp(HP),_).
HP = 130.

8 ?- pokemon(name(Name),_,hp(HP),_), HP > 85,write(Name),nl,fail.
raichu
venusaur
butterfree
charizard
ninetails
poliwrath
blastoise
false.
```

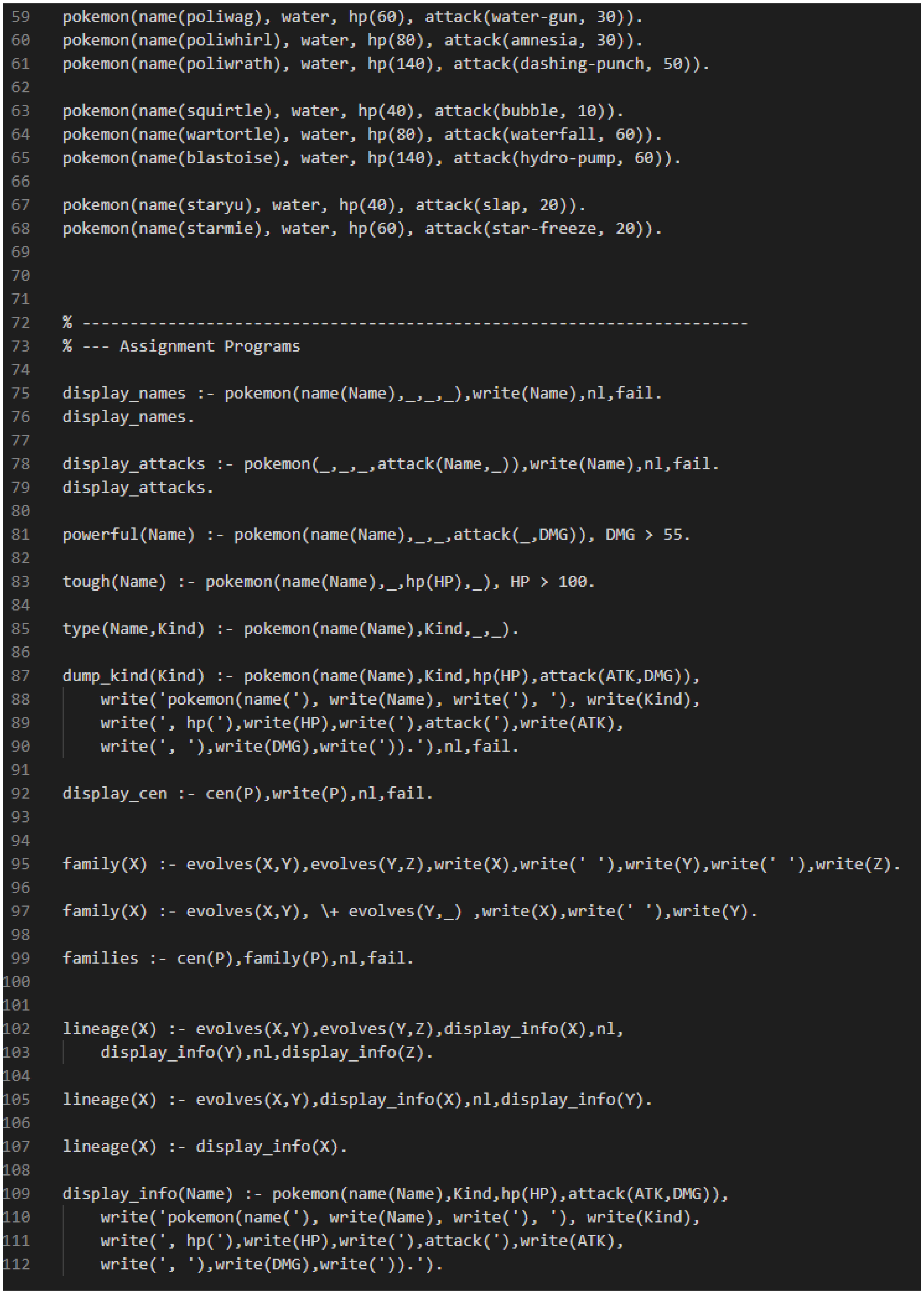


```
11 ?- pokemon(_,_,_,attack(Name,DMG)), DMG > 60,write(Name),nl,fail.
thunder-shock
poison-powder
whirlwind
royal-blaze
fire-blast
false.

12 ?- cen(Name), pokemon(name(Name),_,hp(HP),_), write(Name),write(':  '),write(HP),nl,fail.
pikachu:  60
bulbasaur:  40
caterpie:  50
charmander:  50
vulpix:  60
poliwag:  60
squirtle:  40
staryu:  40
false.
```

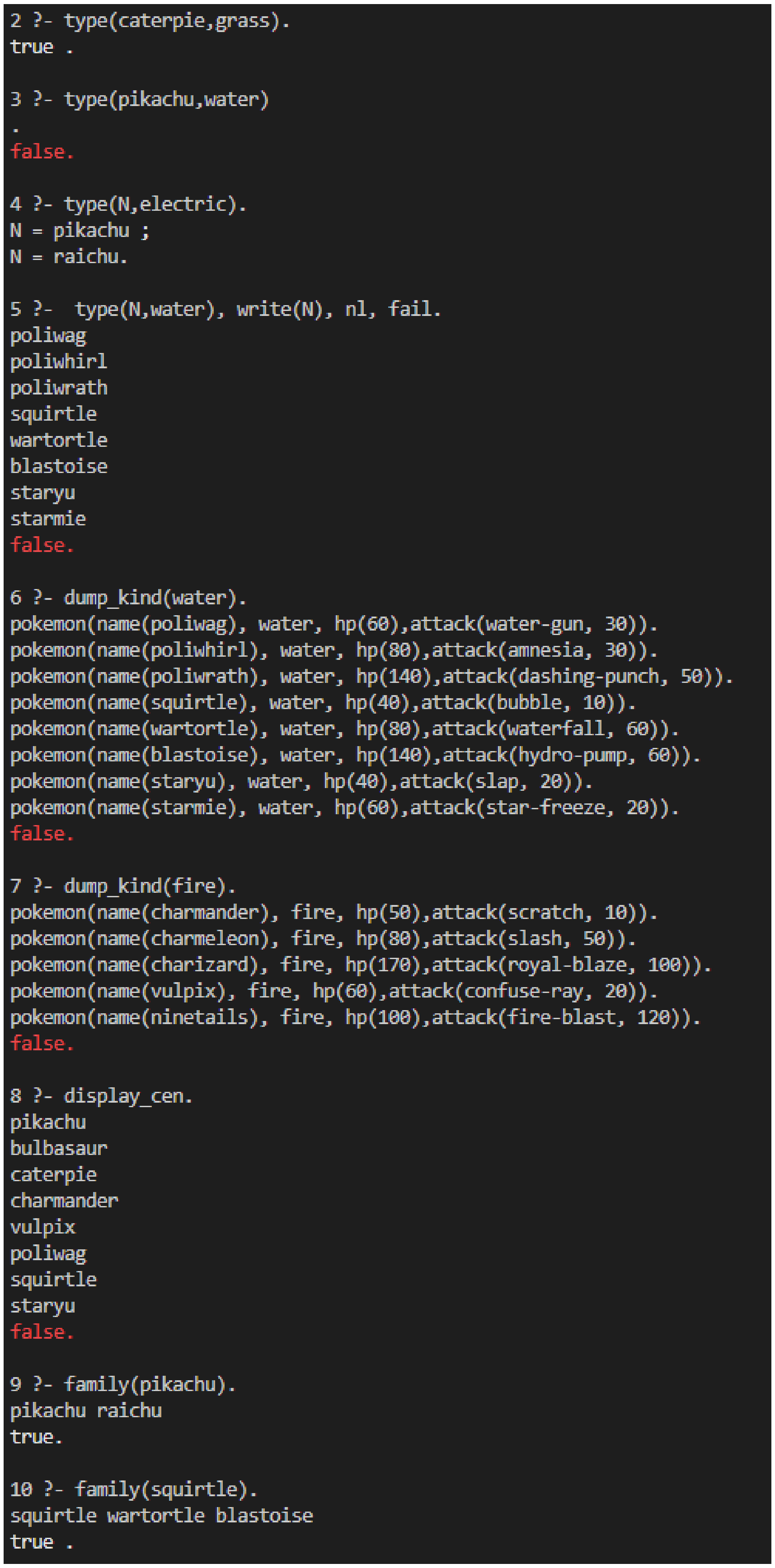


```
% ---------------------------------------------------------------------
% --- cen(P) :: Pokemon P was "creatio ex nihilo"

cen(pikachu).
cen(bulbasaur).
cen(caterpie).
cen(charmander).
cen(vulpix).
cen(poliwag).
cen(squirtle).
cen(staryu).

% ---------------------------------------------------------------------
% --- evolves(P,Q) :: Pokemon P directly evolves to pokemon Q

evolves(pikachu,raichu).
evolves(bulbasaur,ivysaur).
evolves(ivysaur,venusaur).
evolves(caterpie,metapod).
evolves(metapod,butterfree).
evolves(charmander,charmeleon).
evolves(charmeleon,charizard).
evolves(vulpix,ninetails).
evolves(poliwag,poliwhirl).
evolves(poliwhirl,poliwrath).
evolves(squirtle,wartortle).
evolves(wartortle,blastoise).
evolves(staryu,starmie).

% ---------------------------------------------------------------------
% --- pokemon(name(N),T,hp(H),attach(A,D)) :: There is a pokemon with
% --- name N, type T, hit point value H, and attach named A that does
% --- damage D.

pokemon(name(pikachu), electric, hp(60), attack(gnaw, 10)).
pokemon(name(raichu), electric, hp(90), attack(thunder-shock, 90)).

pokemon(name(bulbasaur), grass, hp(40), attack(leech-seed, 20)).
pokemon(name(ivysaur), grass, hp(60), attack(vine-whip, 30)).
pokemon(name(venusaur), grass, hp(140), attack(poison-powder, 70)).

pokemon(name(caterpie), grass, hp(50), attack(gnaw, 20)).
pokemon(name(metapod), grass, hp(70), attack(stun-spore, 20)).
pokemon(name(butterfree), grass, hp(130), attack(whirlwind, 80)).

pokemon(name(charmander), fire, hp(50), attack(scratch, 10)).
pokemon(name(charmeleon), fire, hp(80), attack(slash, 50)).
pokemon(name(charizard), fire, hp(170), attack(royal-blaze, 100)).

pokemon(name(vulpix), fire, hp(60), attack(confuse-ray, 20)).
pokemon(name(ninetails), fire, hp(100), attack(fire-blast, 120)).
```

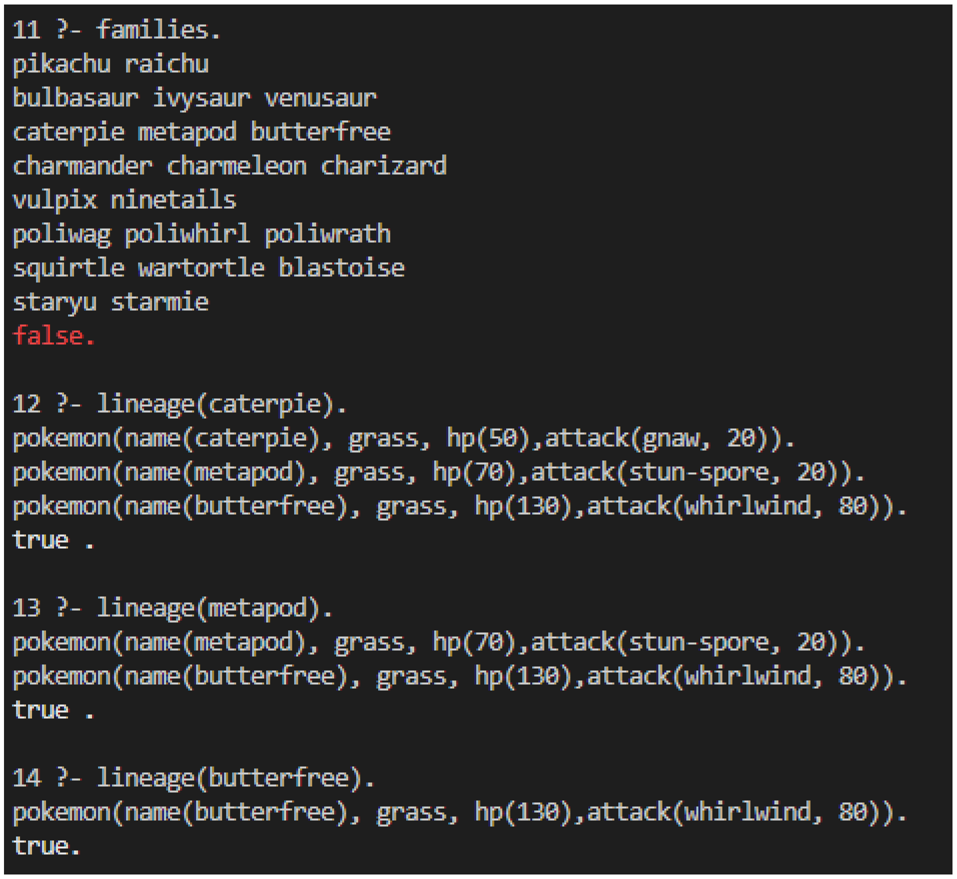
Source code



```
pokemon(name(poliwag), water, hp(60), attack(water-gun, 30)).
pokemon(name(poliwhirl), water, hp(80), attack(amnesia, 30)).
pokemon(name(poliwrath), water, hp(140), attack(dashing-punch, 50)).

pokemon(name(squirtle), water, hp(40), attack(bubble, 10)).
pokemon(name(wartortle), water, hp(80), attack(waterfall, 60)).
pokemon(name(blastoise), water, hp(140), attack(hydro-pump, 60)).

pokemon(name(staryu), water, hp(40), attack(slap, 20)).
pokemon(name(starmie), water, hp(60), attack(star-freeze, 20)).



% ---------------------------------------------------------------------
% --- Assignment Programs

display_names :- pokemon(name(Name),_,_,_),write(Name),nl,fail.
display_names.

display_attacks :- pokemon(_,_,_,attack(Name,_)),write(Name),nl,fail.
display_attacks.

powerful(Name) :- pokemon(name(Name),_,_,attack(_,DMG)), DMG > 55.

tough(Name) :- pokemon(name(Name),_,hp(HP),_), HP > 100.

type(Name,Kind) :- pokemon(name(Name),Kind,_,_).

dump_kind(Kind) :- pokemon(name(Name),Kind,hp(HP),attack(ATK,DMG)),
    write('pokemon(name('), write(Name), write('), '), write(Kind),
    write(', hp('),write(HP),write('),attack('),write(ATK),
    write(', '),write(DMG),write(')).'),nl,fail.

display_cen :- cen(P),write(P),nl,fail.


family(X) :- evolves(X,Y),evolves(Y,Z),write(X),write(' '),write(Y),write(' '),write(Z).

family(X) :- evolves(X,Y), \+ evolves(Y,_) ,write(X),write(' '),write(Y).

families :- cen(P),family(P),nl,fail.


lineage(X) :- evolves(X,Y),evolves(Y,Z),display_info(X),nl,
    display_info(Y),nl,display_info(Z).

lineage(X) :- evolves(X,Y),display_info(X),nl,display_info(Y).

lineage(X) :- display_info(X).

display_info(Name) :- pokemon(name(Name),Kind,hp(HP),attack(ATK,DMG)),
    write('pokemon(name('), write(Name), write('), '), write(Kind),
    write(', hp('),write(HP),write('),attack('),write(ATK),
    write(', '),write(DMG),write(')).').
```



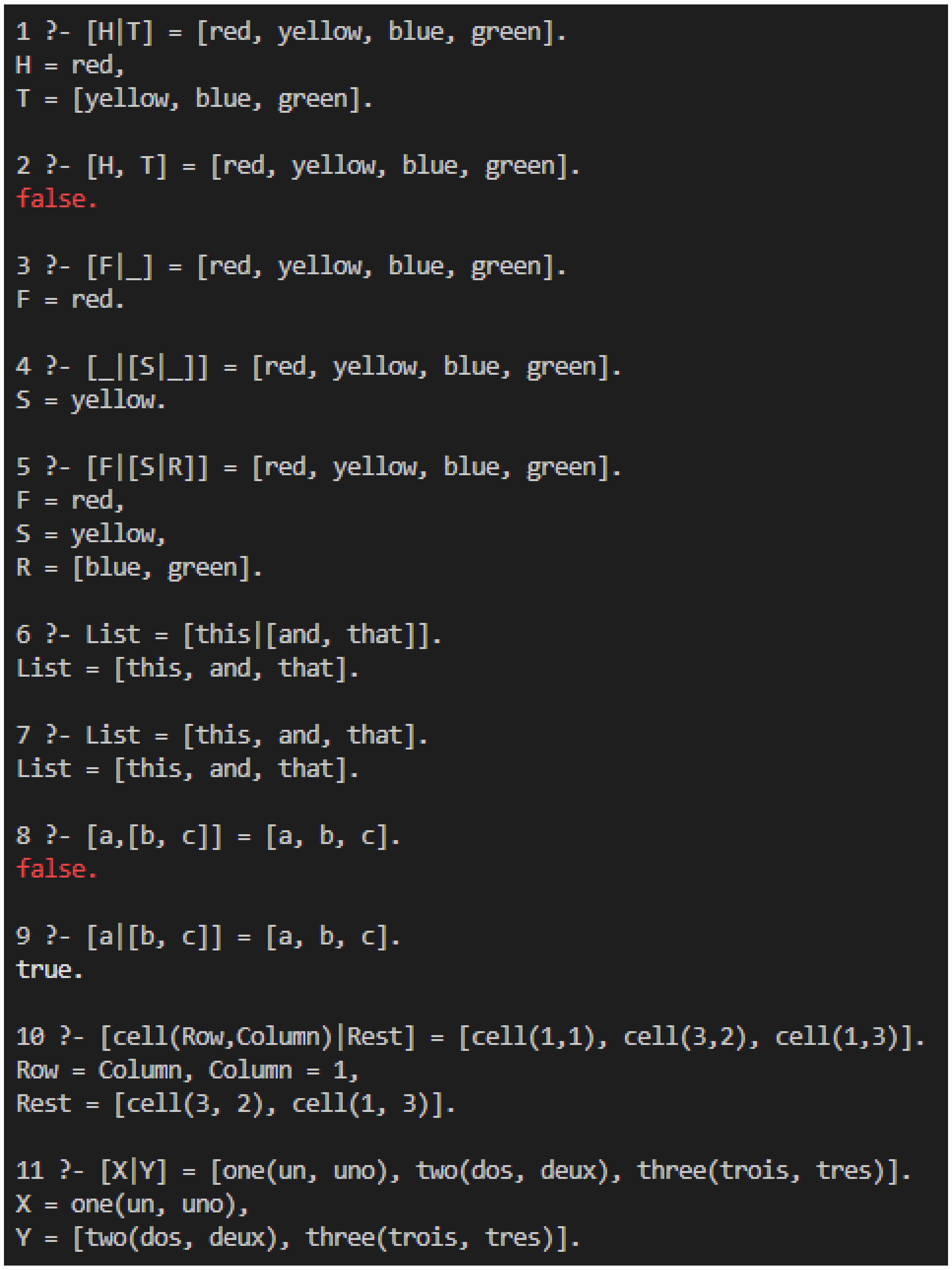
```
2 ?- type(caterpie,grass).
true .

3 ?- type(pikachu,water)
.
false.

4 ?- type(N,electric).
N = pikachu ;
N = raichu.

5 ?-  type(N,water), write(N), nl, fail.
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

6 ?- dump_kind(water).
pokemon(name(poliwag), water, hp(60),attack(water-gun, 30)).
pokemon(name(poliwhirl), water, hp(80),attack(amnesia, 30)).
pokemon(name(poliwrath), water, hp(140),attack(dashing-punch, 50)).
pokemon(name(squirtle), water, hp(40),attack(bubble, 10)).
pokemon(name(wartortle), water, hp(80),attack(waterfall, 60)).
pokemon(name(blastoise), water, hp(140),attack(hydro-pump, 60)).
pokemon(name(staryu), water, hp(40),attack(slap, 20)).
pokemon(name(starmie), water, hp(60),attack(star-freeze, 20)).
false.

7 ?- dump_kind(fire).
pokemon(name(charmander), fire, hp(50),attack(scratch, 10)).
pokemon(name(charmeleon), fire, hp(80),attack(slash, 50)).
pokemon(name(charizard), fire, hp(170),attack(royal-blaze, 100)).
pokemon(name(vulpix), fire, hp(60),attack(confuse-ray, 20)).
pokemon(name(ninetails), fire, hp(100),attack(fire-blast, 120)).
false.

8 ?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

9 ?- family(pikachu).
pikachu raichu
true.

10 ?- family(squirtle).
squirtle wartortle blastoise
true .
```



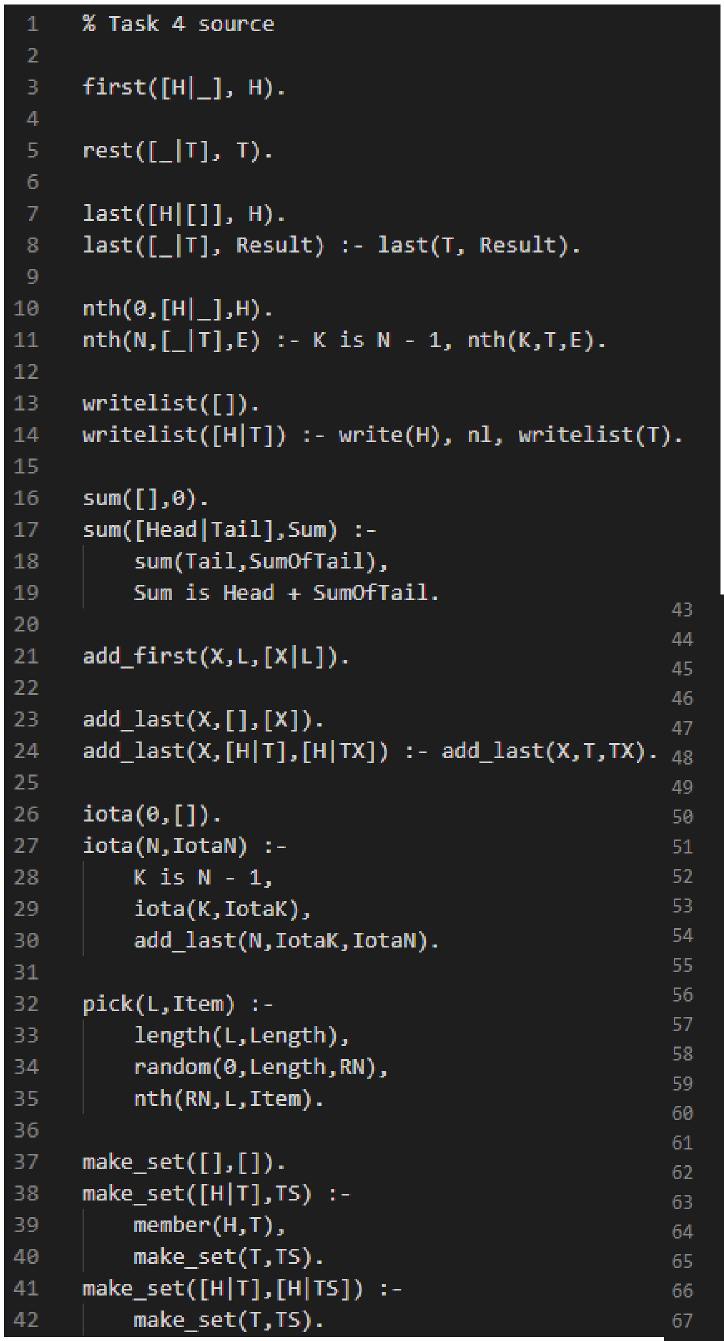
```
11 ?- families.
pikachu raichu
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
staryu starmie
false.

12 ?- lineage(caterpie).
pokemon(name(caterpie), grass, hp(50),attack(gnaw, 20)).
pokemon(name(metapod), grass, hp(70),attack(stun-spore, 20)).
pokemon(name(butterfree), grass, hp(130),attack(whirlwind, 80)).
true .

13 ?- lineage(metapod).
pokemon(name(metapod), grass, hp(70),attack(stun-spore, 20)).
pokemon(name(butterfree), grass, hp(130),attack(whirlwind, 80)).
true .

14 ?- lineage(butterfree).
pokemon(name(butterfree), grass, hp(130),attack(whirlwind, 80)).
true.
```

Task 4 – Lisp Processing in Prolog

Head/Tail Demo

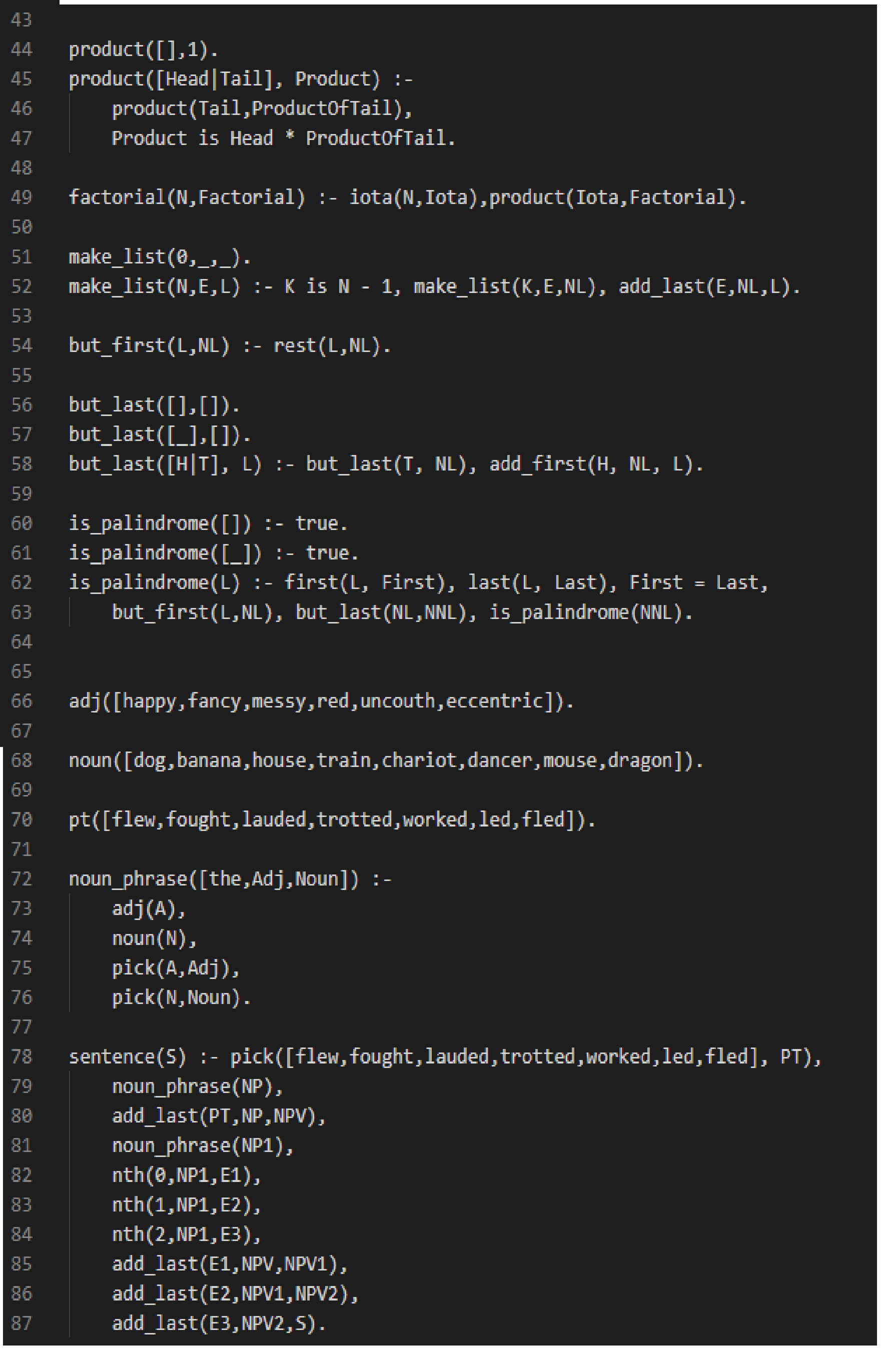


```
1 ?- [H|T] = [red, yellow, blue, green].
H = red,
T = [yellow, blue, green].

2 ?- [H, T] = [red, yellow, blue, green].
false.

3 ?- [F|_] = [red, yellow, blue, green].
F = red.

4 ?- [_|[S|_]] = [red, yellow, blue, green].
S = yellow.

5 ?- [F|[S|R]] = [red, yellow, blue, green].
F = red,
S = yellow,
R = [blue, green].

6 ?- List = [this|[and, that]].
List = [this, and, that].

7 ?- List = [this, and, that].
List = [this, and, that].

8 ?- [a,[b, c]] = [a, b, c].
false.

9 ?- [a|[b, c]] = [a, b, c].
true.

10 ?- [cell(Row,Column)|Rest] = [cell(1,1), cell(3,2), cell(1,3)].
Row = Column, Column = 1,
Rest = [cell(3, 2), cell(1, 3)].

11 ?- [X|Y] = [one(un, uno), two(dos, deux), three(trois, tres)].
X = one(un, uno),
Y = [two(dos, deux), three(trois, tres)].
```

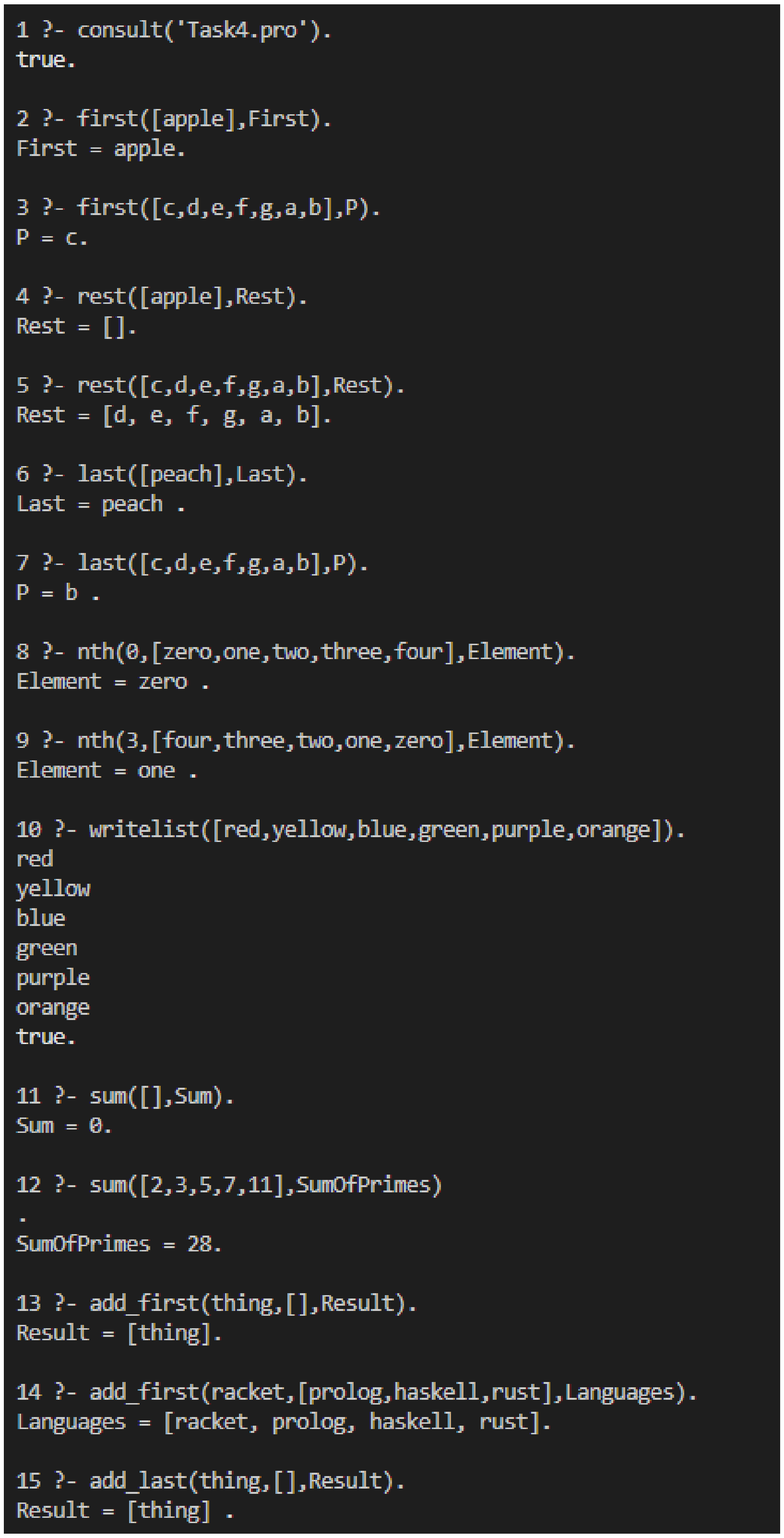


```prolog
% Task 4 source

first([H|_], H).

rest([_|T], T).

last([H|[]], H).
last([_|T], Result) :- last(T, Result).

nth(0,[H|_],H).
nth(N,[_|T],E) :- K is N - 1, nth(K,T,E).

writelist([]).
writelist([H|T]) :- write(H), nl, writelist(T).

sum([],0).
sum([Head|Tail],Sum) :-
    sum(Tail,SumOfTail),
    Sum is Head + SumOfTail.

add_first(X,L,[X|L]).

add_last(X,[],[X]).
add_last(X,[H|T],[H|TX]) :- add_last(X,T,TX).

iota(0,[]).
iota(N,IotaN) :-
    K is N - 1,
    iota(K,IotaK),
    add_last(N,IotaK,IotaN).

pick(L,Item) :-
    length(L,Length),
    random(0,Length,RN),
    nth(RN,L,Item).

make_set([],[]).
make_set([H|T],TS) :-
    member(H,T),
    make_set(T,TS).
make_set([H|T],[H|TS]) :-
    make_set(T,TS).
```

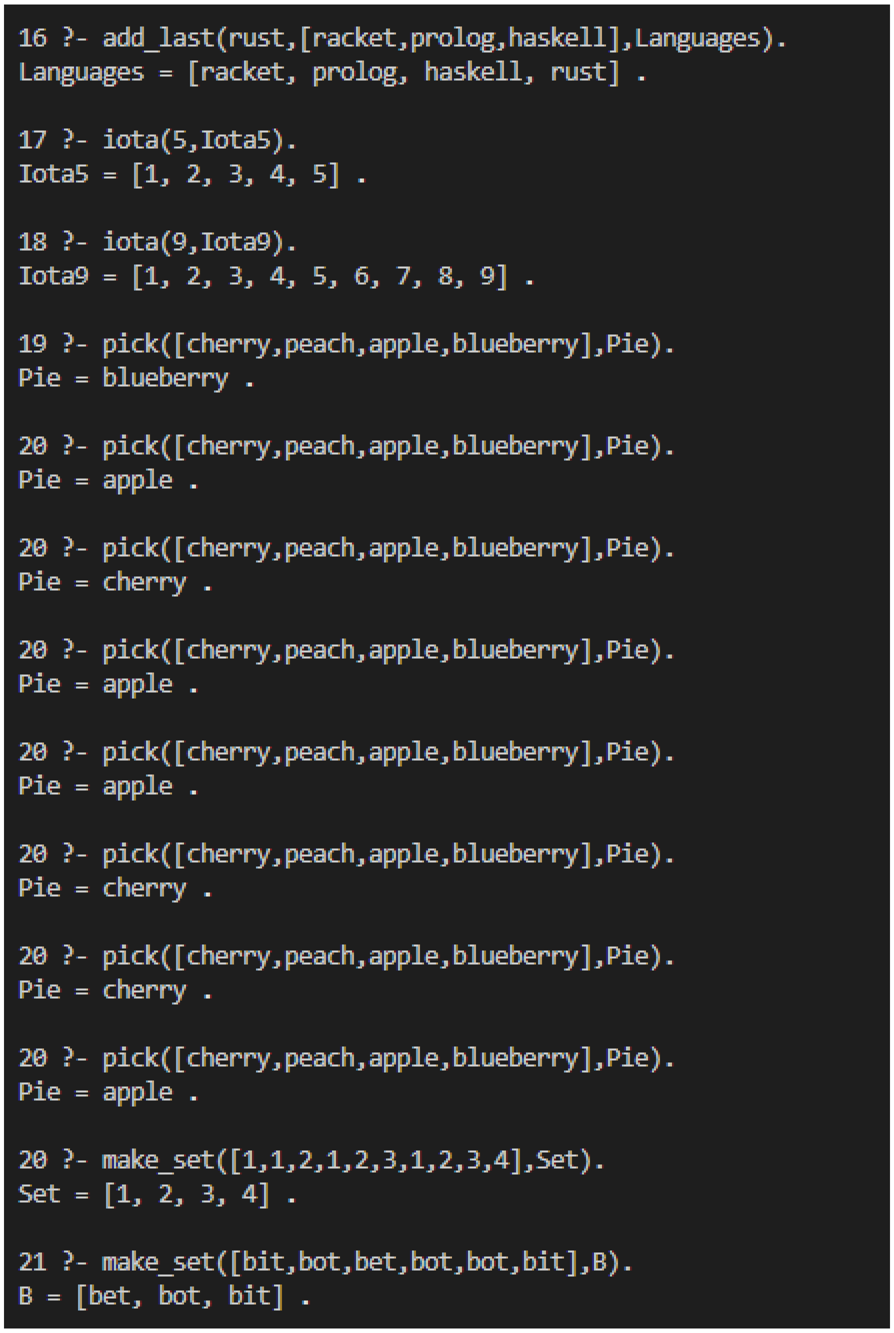
Task 4 KB



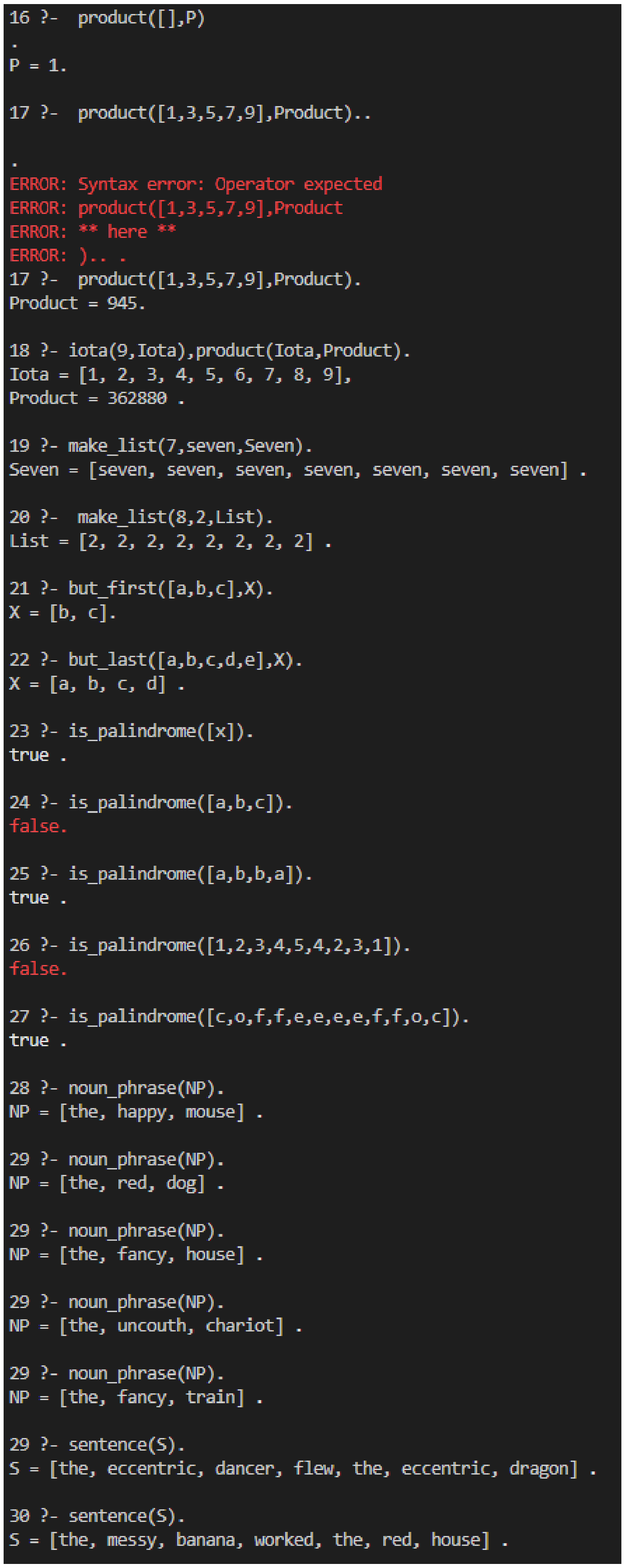
```prolog

product([],1).
product([Head|Tail], Product) :-
    product(Tail,ProductOfTail),
    Product is Head * ProductOfTail.

factorial(N,Factorial) :- iota(N,Iota),product(Iota,Factorial).

make_list(0,_,_).
make_list(N,E,L) :- K is N - 1, make_list(K,E,NL), add_last(E,NL,L).

but_first(L,NL) :- rest(L,NL).

but_last([],[]).
but_last([_],[]).
but_last([H|T], L) :- but_last(T, NL), add_first(H, NL, L).

is_palindrome([]) :- true.
is_palindrome([_]) :- true.
is_palindrome(L) :- first(L, First), last(L, Last), First = Last,
    but_first(L,NL), but_last(NL,NNL), is_palindrome(NNL).


adj([happy,fancy,messy,red,uncouth,eccentric]).

noun([dog,banana,house,train,chariot,dancer,mouse,dragon]).

pt([flew,fought,lauded,trotted,worked,led,fled]).

noun_phrase([the,Adj,Noun]) :-
    adj(A),
    noun(N),
    pick(A,Adj),
    pick(N,Noun).

sentence(S) :- pick([flew,fought,lauded,trotted,worked,led,fled], PT),
    noun_phrase(NP),
    add_last(PT,NP,NPV),
    noun_phrase(NP1),
    nth(0,NP1,E1),
    nth(1,NP1,E2),
    nth(2,NP1,E3),
    add_last(E1,NPV,NPV1),
    add_last(E2,NPV1,NPV2),
    add_last(E3,NPV2,S).
```



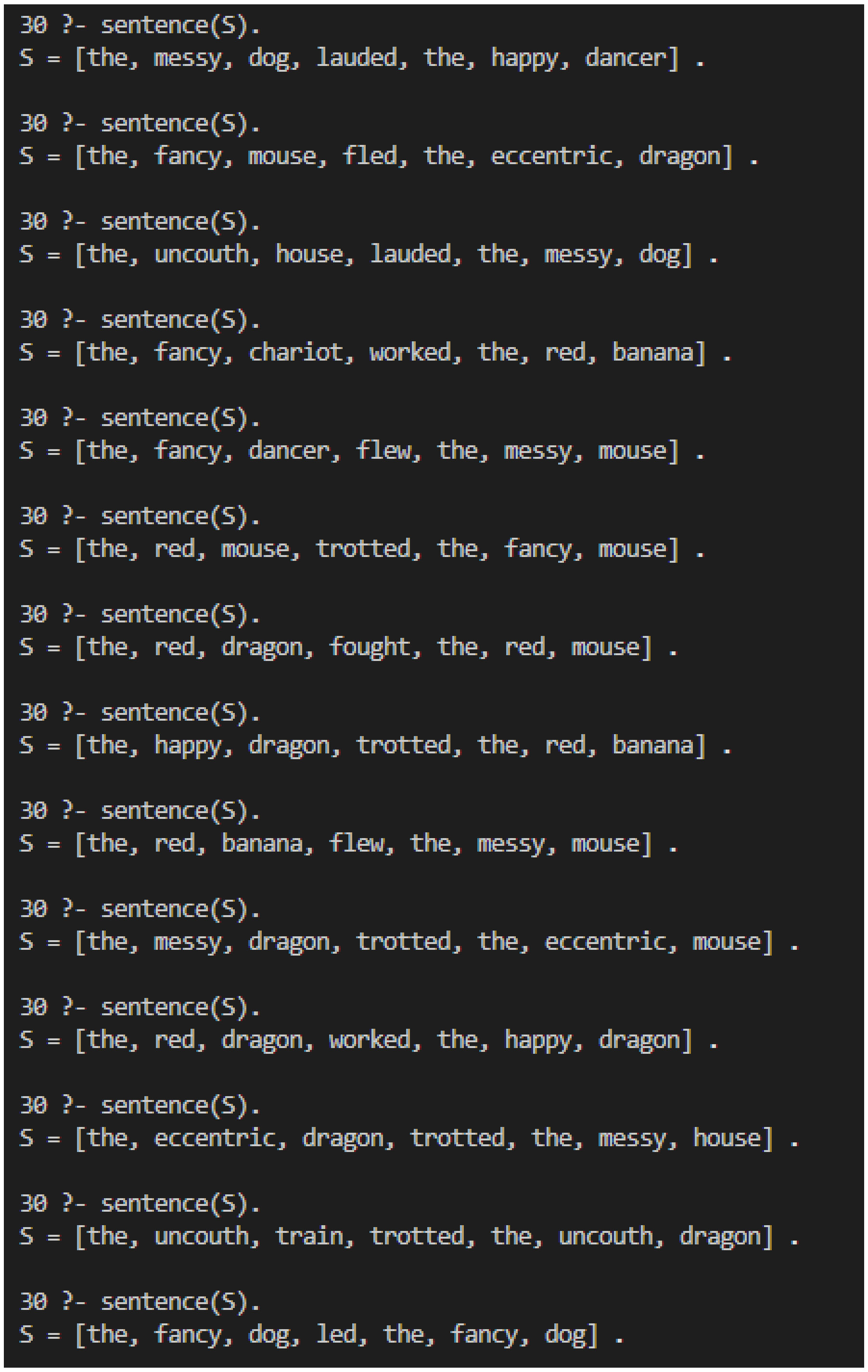
```
1 ?- consult('Task4.pro').
true.

2 ?- first([apple],First).
First = apple.

3 ?- first([c,d,e,f,g,a,b],P).
P = c.

4 ?- rest([apple],Rest).
Rest = [].

5 ?- rest([c,d,e,f,g,a,b],Rest).
Rest = [d, e, f, g, a, b].

6 ?- last([peach],Last).
Last = peach .

7 ?- last([c,d,e,f,g,a,b],P).
P = b .

8 ?- nth(0,[zero,one,two,three,four],Element).
Element = zero .

9 ?- nth(3,[four,three,two,one,zero],Element).
Element = one .

10 ?- writelist([red,yellow,blue,green,purple,orange]).
red
yellow
blue
green
purple
orange
true.

11 ?- sum([],Sum).
Sum = 0.

12 ?- sum([2,3,5,7,11],SumOfPrimes)
.
SumOfPrimes = 28.

13 ?- add_first(thing,[],Result).
Result = [thing].

14 ?- add_first(racket,[prolog,haskell,rust],Languages).
Languages = [racket, prolog, haskell, rust].

15 ?- add_last(thing,[],Result).
Result = [thing] .
```



```
16 ?- add_last(rust,[racket,prolog,haskell],Languages).
Languages = [racket, prolog, haskell, rust] .

17 ?- iota(5,Iota5).
Iota5 = [1, 2, 3, 4, 5] .

18 ?- iota(9,Iota9).
Iota9 = [1, 2, 3, 4, 5, 6, 7, 8, 9] .

19 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = blueberry .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = apple .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = apple .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = apple .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = cherry .

20 ?- pick([cherry,peach,apple,blueberry],Pie).
Pie = apple .

20 ?- make_set([1,1,2,1,2,3,1,2,3,4],Set).
Set = [1, 2, 3, 4] .

21 ?- make_set([bit,bot,bet,bot,bot,bit],B).
B = [bet, bot, bit] .
```



```
16 ?-  product([],P)
.
P = 1.

17 ?-  product([1,3,5,7,9],Product)..


.
ERROR: Syntax error: Operator expected
ERROR: product([1,3,5,7,9],Product
ERROR: ** here **
ERROR: ).. .
17 ?-  product([1,3,5,7,9],Product).
Product = 945.

18 ?- iota(9,Iota),product(Iota,Product).
Iota = [1, 2, 3, 4, 5, 6, 7, 8, 9],
Product = 362880 .

19 ?- make_list(7,seven,Seven).
Seven = [seven, seven, seven, seven, seven, seven, seven] .

20 ?-  make_list(8,2,List).
List = [2, 2, 2, 2, 2, 2, 2, 2] .

21 ?- but_first([a,b,c],X).
X = [b, c].

22 ?- but_last([a,b,c,d,e],X).
X = [a, b, c, d] .

23 ?- is_palindrome([x]).
true .

24 ?- is_palindrome([a,b,c]).
false.

25 ?- is_palindrome([a,b,b,a]).
true .

26 ?- is_palindrome([1,2,3,4,5,4,2,3,1]).
false.

27 ?- is_palindrome([c,o,f,f,e,e,e,e,f,f,o,c]).
true .

28 ?- noun_phrase(NP).
NP = [the, happy, mouse] .

29 ?- noun_phrase(NP).
NP = [the, red, dog] .

29 ?- noun_phrase(NP).
NP = [the, fancy, house] .

29 ?- noun_phrase(NP).
NP = [the, uncouth, chariot] .

29 ?- noun_phrase(NP).
NP = [the, fancy, train] .

29 ?- sentence(S).
S = [the, eccentric, dancer, flew, the, eccentric, dragon] .

30 ?- sentence(S).
S = [the, messy, banana, worked, the, red, house] .
```



```
30 ?- sentence(S).
S = [the, messy, dog, lauded, the, happy, dancer] .

30 ?- sentence(S).
S = [the, fancy, mouse, fled, the, eccentric, dragon] .

30 ?- sentence(S).
S = [the, uncouth, house, lauded, the, messy, dog] .

30 ?- sentence(S).
S = [the, fancy, chariot, worked, the, red, banana] .

30 ?- sentence(S).
S = [the, fancy, dancer, flew, the, messy, mouse] .

30 ?- sentence(S).
S = [the, red, mouse, trotted, the, fancy, mouse] .

30 ?- sentence(S).
S = [the, red, dragon, fought, the, red, mouse] .

30 ?- sentence(S).
S = [the, happy, dragon, trotted, the, red, banana] .

30 ?- sentence(S).
S = [the, red, banana, flew, the, messy, mouse] .

30 ?- sentence(S).
S = [the, messy, dragon, trotted, the, eccentric, mouse] .

30 ?- sentence(S).
S = [the, red, dragon, worked, the, happy, dragon] .

30 ?- sentence(S).
S = [the, eccentric, dragon, trotted, the, messy, house] .

30 ?- sentence(S).
S = [the, uncouth, train, trotted, the, uncouth, dragon] .

30 ?- sentence(S).
S = [the, fancy, dog, led, the, fancy, dog] .
```